Learning Long-Term Visual Dynamics With Region Proposal Interaction Networks

Jack Chuang, Moses Prasad Varghese
University of Washington
Seattle, WA 98195
{erhchc, mosespv}@uw.edu

1 Introduction

1.1 Abstract

There are several methods proposed to build physical common-sense. One such example is leveraging deep networks to build data-driven models of intuitive physics[2], and another one is to leverage graph neural networks to capture interactions of the entities in the scenes[3]. An approach that is slightly similar to our paper is interaction models that uses pixels for scaling and object interaction for reasoning[4]. However, these methods have certain limitations. Continuous requirements of the ground truth state space implying they don't scale to long-term predictions due to the lack of reasoning. Another draw back is that these model deal with a fixed number and order of object features which reduces the accuracy of their prediction. In this paper, the authors focus on deriving a long-term aspect of prediction by just considering environments. The objective is to model long-term interactions over time to plan successfully for new instances, and to work from raw visual input in complex real-world environments. With these goal in mind, they propose Region Proposal Interaction Network (RPIN), which comes in with two main parts. First is the RoIPooling[5] which extracts the information of the object itself as well as the environment around it. second is the extension of the Interaction Network[6] —- Convolutional Interaction Networks that perform interaction reasoning on the extracted RoI features.

1.2 Motivation

Standard deep neural network building blocks is used in general IN(Interaction Network), such as multilayer perceptrons (MLP), matrix operations, etc., which can be trained efficiently from data using gradient-based optimisation, such as stochastic gradient descent. And in this paper, a combination of convolution and IN is presented for dealing with nonlinear relations and dynamics. This approach proposed by the authors is simple, yet outperforms the state-of-the-art methods in both simulation and real datasets. The method reduces the prediction error by 75% in the complex PHYRE(PHYsical REasoning) environment and achieves state-of-the-art performance on the PHYRE benchmark in both within task generalization (PHYRE-W), where same group of objects used in both training and testing are being trained in different settings from the testing environment, and cross task generalization (PHYRE-C), where the settings used for testing and objects are never present during training.

2 Description of Datasets and Hyperparameter Selection

We use four different sets of data: PHYRE, ShapeStacks(SS), Real World Billiards (RealB) and Simulation Billiards (SimB) to evaluate our model's prediction performance and ability for physical reasoning and planning.

2.1 PHYRE

A new benchmark to assess an AI agent's capacity for physical reasoning. Inspired by popular physics-puzzle games, we developed PHYRE (the name refers to PHYsical REasoning) to include multiple tasks that are simple for humans but daunting for current AI techniques and in the paper the BALL-tier of the PHYRE benchmark was used [9]. Since we have two types of generalization as explained above, for within task generalization (PHYRE-W), the dataset for training contains 80 templates for each of the 25 task. The testing set contains the remaining 20 templates from each task. For cross task generalization (PHYRE-C), the training set contains 100 templates from 20 tasks while the test set contains 100 templates from the remaining 5 tasks. In order to collect the trajectories to train our model, for each template, a randomly sampled maximum 100 success and 400 failure actions are gathered and the image sequence is temporally downsampled by 60 for faster inference. But for the physical reasoning and planning task experiments, the model is trained with 400 successful actions and 1600 failure actions per template in order for proper learning. Since we split the dataset to training and validation, the validation set is used to select a appropriate hyper-parameters first, and then training was done on the union of train and validation sets and reported the final results in the test set.

The image is resized to 128×128. We train the model for 150000 iterations with a learning rate $2 \times 10{-}4$, weight decay 3×10^{-7} and batch size 20.

2.2 ShapeStacks(SS)

This is a dataset that contains many stacked objects such as cube, cylinder or balls [10]. There are 1,320 training videos and 296 testing videos, with 32 frames per video and specifically objects' center positions are provided. From the analysis of the paper [10] we assume the object bounding box is square and of size 70*70. This dataset contains many stacked objects and we set T = 15 were T is the future timesteps. Uniquely for this dataset, the concept of uncertainty estimation is also introduced [10]. In this concept, we model a latent distribution using a variational auto-encoder [11] were first we develop an encoder say, 'h' which takes the image feature from first F^0 and last frame F^T of a video sequence as the input. The output of h is a distribution parameter, denoted by h(u|F^0, F^T). Given a particular sample from such distribution, we recover the latent variable by feeding them into a one-layer LSTM and merge into the object feature x_i^t for ith object and t timestep. In this paper, the SS dataset model is trained with an additional loss that minimize the KL divergence between the predicted distribution and normal distribution. During reasoning, a sample of 100 trajectories for each test image was taken and reported the minimum of them. The image is resized to 224×224. We train the model for 25000 iterations with a learning rate 2×10^{-4} no weight decay, and batch size 40. We take loss weight of KL-divergence as 3×10^{-5}. During inference, we randomly sample 100 outputs from our model, and select the best (in terms of the distance to ground-truth) of them as our model's output.

2.3 Real World Billiards (RealB)

This dataset was created by downloading the "Three-cushion Billiards" videos from professional games with different viewpoints from Youtube. There are 62 training videos with 18306 frames, and 5 testing videos with 1995 frames. The bounding box annotations are from ResNet-101 FPN detector [12] pretrained on COCO [13] and fine-tuned on a subset of 30 images from RealB dataset. Here, T was set to 20. It was found that the bounding box prediction results were accurate enough to serve as the groundtruth. After running the detector, images with incorrect detections were filtered out with manual inspection of the dataset. The image is resized to 192×64. We train the model for 240000 iterations with a learning rate 1×10^{-4}, weight decay 1×10^{-6} and batch size 20.

2.4 Simulation Billiards (SimB)

As the name suggests, a simulated billiard environment with three different colored balls with a radius 2 randomly placed in a 64×64 image was developed as the dataset. At starting point, one ball is moving with a randomly sampled velocity. We generate 1,000 video sequences for training and 1,000 video sequences for testing, with 100 frames per sequence. We will also evaluate the ability to generalize to more balls and different sized balls in the experiment section. In this dataset, we set T = 20. To get the initial velocity, the magnitude (number of pixels moved per timestep) is sampled from [2, 3, 4, 5, 6] and the direction is sampled from $[6i\pi, i = 0, 1, \ldots 11]$. We train the model for 100000 iterations with a learning rate 2×10^{-3}, weight decay 1×10^{-6}, and batch size 200.

2.5 Common analysis of dataset and parameters

For PHYRE and SS, it is possible to infer the future from just the initial configuration. So we set input number of video frames to be N = 1 in these two datasets. For SimB and RealB, the objects are moving in a flat table so we cannot infer the future trajectories based on a single image. Therefore in this setting, we set N = 4 to infer the velocity/acceleration of objects. We set number of previous timesteps, k = N in all the dataset because we only have access to N features when we make prediction at t = N + 1. We predict object masks for PHYRE and ShapeStacks, and only predict bounding boxes for Billiard datasets. We use Adam optimizer [14] with cosine decay [15] to train our networks. We set 'd' to be 256 except for simulation billiard 'd' is 64. During training, T (denoted as T_{train}) is set to be 20 for SimB and RealB, 5 for PHYRE, and 15 for SS. The discounted factor λ_t is set to be $(\frac{current\_iter}{max\_iter})^t$.

3 Model Description, Baselines and Methodology

Our model takes N video frames and the corresponding object bounding boxes as inputs, and outputs the objects' bounding boxes and masks for the future T timesteps and the overall model is shown below:

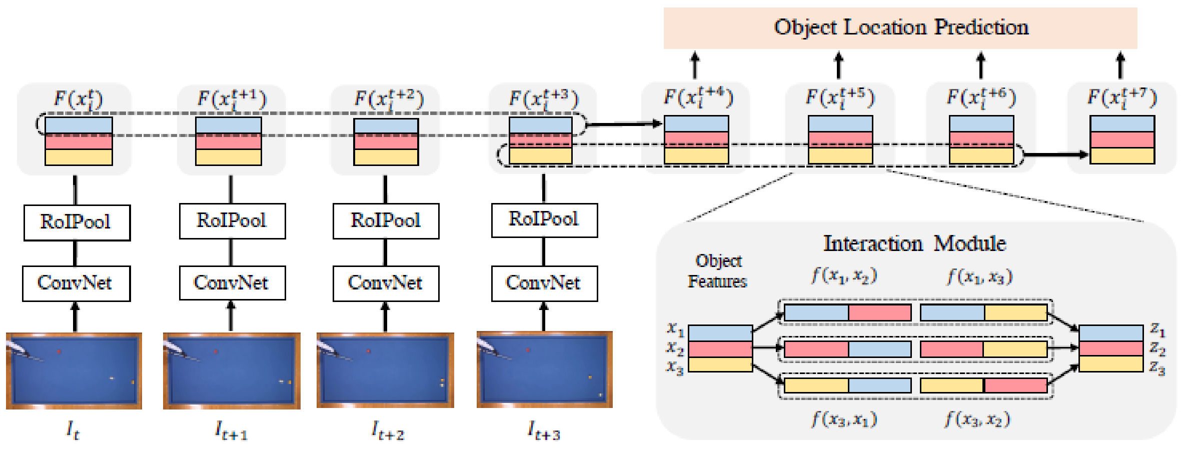

Figure 1: Overview of the RPIN model

For each frame, we first extract the image features using a ConvNet. Then we apply RoIPooling [5] to obtain the object-centric visual features. These features are then forwarded to our Convolutional

Interaction Networks (CIN) to perform objects' interaction reasoning and used to predict future object bounding boxes and masks. The whole pipeline is trained end-to-end by minimizing the loss between the predicted outputs and the ground-truth.

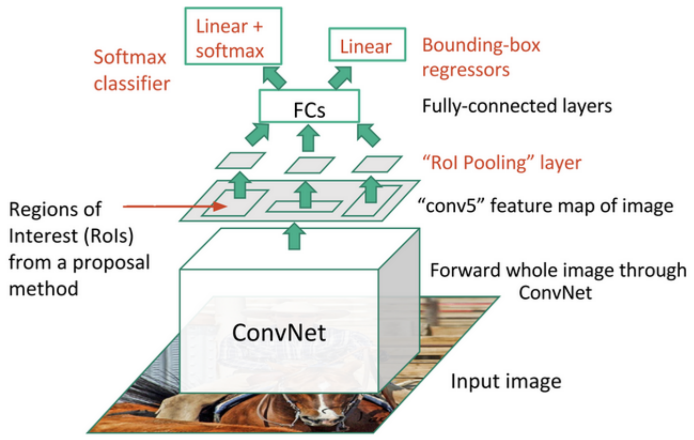

Figure 2: Steps of doing ROI pooling

3.1 Main Model: Object Centric Representation

We apply the hourglass network [19] to extract the image features. Given an input RGB image $I \in R^{3\times H\times W}$, the hourglass network firstly downsample the input via one 2-strided convolution and 2-strided max pooling layers, and then refine the representation by a U-Net-like encoder-decoder modules [20]. The hourglass network provides features with a large receptive field and a fine spatial resolution (4 times smaller than the input image), both of which are crucial to accurately model object-object and object-environment interactions. We denote the number of output channels of the feature map as 'd'. On top of this feature map, we use RoIPooling as explained above to extract a $d \times h \times w$ object-centric features. RoIPooling takes the feature map and a object bounding box as input. The region corresponding to the bounding box on the feature map is cropped and resize to a fixed spatial size (denoted as $h \times w$). We use x_i^t to represent the feature at t-th timestep for the i-th object. Such feature representation differs from previous method in two perspectives: 1) It is extracted from image features with a large receptive field, which gives plenty of context information around the object. 2) The feature representation is a 3-dimensional feature map rather than a single vector representation. It can represent the objects' shapes while the vector representation cannot because the spatial dimension is flattened.

3.2 Convolutional Interaction Networks

The original interaction network (IN) [16, 18] is a general-purpose data-driven method to model and predict physical dynamics. It takes the feature vectors of m objects at timestep $t : X = \left\{x_1^t, x_2^t, \ldots, x_m^t \mid x_i^t \in \mathbb{R}^d\right\}$ and performs object reasoning f_O as well as relational reasoning f_R on these features. Specifically, the updated rule of object features can be described as:

$$e_i^t = f_A\left(f_O\left(x_i^t\right) + \sum_{j\neq i} f_R\left(x_i^t, x_j^t\right)\right),$$

$$z_i^t = f_Z\left(x_i^t, e_i^t\right),$$

$$x_i^{t+1} = f_P\left(z_i^t, z_i^{t-1}, \ldots, z_i^{t-k}\right).$$

In the above equation, f_A is the function to calculate the effect of both of object reasoning and relational reasoning results. And f_Z is used to combine the original object state and the reasoning effect. Finally, f_P is used to do future state predictions based on one or more previous object states. In IN, $f_{O,R,A,Z,P}$ are instantiated by a fully-connected layer with learnable weights.

Convolutional Interaction Networks (CIN). The input of CIN is m object feature maps at timestep $t:$ $X = \left\{x_1^t, x_2^t, \ldots, x_m^t \mid x_i^t \in \mathbb{R}^{d\times h\times w}\right\}$. The high-level update rule is the same as IN, but the key difference is that we use convolution to instantiate $f_{O,R,A,Z,P}$. Such instantiation is crucial to utilize the spatial information encoded in our object feature map and to effectively reason future object states. Specifically, we have

$$f_R\left(x_i^t, x_j^t\right) = W_R * \left[x_i^t, x_j^t\right] \quad f_O\left(x_i^t\right) = W_O * x_i^t$$
$$f_A\left(x_i^t\right) = W_A^T * x_i^t \quad f_Z\left(x_i^t, e_i^t\right) = W_Z * \left[x_i^t, e_i^t\right]$$
$$f_P\left(z_i^t, z_i^{t-1}, \ldots, z_i^{t-k}\right) = W_P * \left[z_i^t, z_i^{t-1}, \ldots, z_i^{t-k}\right]$$

In the above equations, * denotes the convolution operator, and $[\cdot,\cdot]$ denotes concatenation along the channel dimension. $W_R, W_Z \in \mathbb{R}^{d\times 2d\times 3\times 3}, W_O, W_A \in \mathbb{R}^{d\times d\times 3\times 3}, W_P \in \mathbb{R}^{d\times (kd)\times 3\times 3}$ are learnable weights of the convolution kernels with kernel size 3×3. We also add ReLU activation after each of the convolutional operators.

3.3 Regional Proposal Interactive Network - Learning

Our model predicts the future bounding box and (optionally) masks of each object. Given the predicted feature x_i^{t+1}, we use a simple two layer MLP decoder to estimate its bounding boxes coordinates and masks. The bounding box decoder takes a flattened object feature map as input. It firstly projects it to d dimensional vector, and outputs 4-d vector, representing the center location and size of the box. The mask decoder is of the same architecture but has 21×21 output channels, representing a 21×21 binary masks inside the corresponding bounding boxes. We use the ℓ^2 loss for bounding box predictions. For mask prediction, we use spatial cross-entropy loss which sums the cross-entropy values of a 21×21 predicted positions. The objective can be written as:

$$L_p = \sum_{t=1}^{T} \lambda_t \sum_{i=1}^{n} \left(\left\|\hat{B}_i^{t+1} - B_i^{t+1}\right\|_2^2 + CE\left(\hat{M}_i^{t+1}, M_i^{t+1}\right)\right)$$

We use discounted loss during training [16] to mitigate the effect of inaccurate prediction at early training stage and λ_t is the discounted factor.

3.4 Network Architecture, Planning and Baseline models

3.4.1 Network Architecture

We use the same backbone network for our method and all the baselines which are VIN, OM and CVP. We choose the hourglass network [19] as the image feature extractor. Given an input image, the hourglass network firstly applies a 7 stride-2 convolution, three residual blocks with channel dimension d=4, and a stride-2 max pooling on it. Then this intermediate feature representation is fed into one hourglass modules. In the hourglass module, the feature maps are downsampled with 3 stride-2 residual blocks and then up-sampled with nearest neighbor interpolation. The dimensions of both the input channel and the output channel of each residual block are d. For SimB, we use d = 64 since the visual information in this environment is relatively simple. For RealB, PHYRE, and ShapeStacks, d = 256. We use batch normalization before each convolutional layer in the backbone network. The resulting feature map size is $d \times \frac{H}{4} \times \frac{W}{4}$ where H, W is the input image size and 4 is the spatial feature stride of the hourglass network. The output features are transformed to object-centric representations differently for our method and baseline models. There are a large amount of work studying how to estimate objects' states and predict their dynamics from raw image inputs. They fall into the three following categories:

VIN Instead of using object-centric spatial pooling to extract object features, it use a ConvNet to globally encode an image to a fixed $d \times m$ dimensional vector. Different channels of the vector is assigned to different objects [16]. This approach requires specifying a fixed number of objects and a fixed mapping between feature channels and object identity, which make it impossible to generalize to different number of objects and different appearances. The resulting feature map from our architecture is forwarded to 4 convolutional layers with stride 2 and a spatial global average pooling layer to get a d dimensional feature vector. This feature vector is transformed to $d \times m$ by one fully-connected layer. This feature will be reshaped to m vectors with d channel dimensional to represent each object. This representation is refined by two $d \times d$ fully-connected layers and passed to the (convolutional) interaction network.

Object Masking (OM) This approach takes one image and m object bounding boxes or masks as input [17]. For each proposal, only the pixels inside object proposals are kept while others are set to 0, leading to m masked images. This approach assumes no background information is needed thus fails to predict accurate trajectories in complex environments such as PHYRE. And it also cost m times computational resources. Here, the output feature map is of shape $m \times d \times \frac{H}{4}\frac{W}{4}$ because this method produce m masked (or cropped) images as input. The resulting feature map is forwarded to 4 convolutional layers with stride 2 and a spatial global average pooling layer to get a d dimensional feature vector and the output feature is of size $m \times d$, representing the features of m objects. This representation is refined by two $d \times d$ fully-connected layers and passed to the (convolutional) interaction network.

CVP The object feature is extracted by cropping the object image patch and forwarding it to an encoder [10]. Since the object features are directly extracted from the raw image patches, the context information is also ignored. The CVP's feature extraction method was re-implemented within the model framework. This baseline model is also changed to object-centric representation similar to the OM baseline model

3.4.2 Planning Model

We apply planning model only to PHYRE and SimB dataset. Given an initial state (represented by an image) and a goal, we aim to produce an action that can lead to the goal from the initial state. Our planning algorithm works in a similar way as visual imagination [21]: Firstly, we select a candidate action 'a' from a candidate action set A. Then we generate the input images I. For SimB, we train a forward model take the initial image and the action embedding to generate object positions for the next 3 steps. Then we use the simulator to generate the 3 images. For PHYRE, we convert the action to the red ball using the simulator and get the initial image. After that, we run our model described in results section of planning actions and the score of each action is simply the classifier's output. We then select the action with the max score. We introduce the action set for each task in the following section and how to design distance function in subsequently.

Action Set For simulation billiard, the action is 3 dimensional. The first two dimensions stand for the direction of the force. The last dimension stands for the magnitude of the force. During doing planning, we enumerate over 5 different magnitudes and 12 different angles, leading to 60 possible actions. All of the initial condition is guaranteed to have a solution. For PHYRE, the action is also 3 dimensional. The first two dimensions stand for the location placing the red ball. The last dimension stands for the radius of the ball. Following [9], we use the first 10000 actions provided by the benchmark.

Distance Function Init-End State Error. Denote the given target location of m objects as $y \in R^{m\times 2}$. We use the following distance function, which measures the distance between the final rollout location and the target location:

$D = \sum_{i=1}^{m} \sum_{j=1}^{2} (\hat{p}_{T,i,j} - y_{i,j})^2$

Hitting Accuracy. Denote the given initial location of m objects as $x \in R^{m\times 2}$. We apply force at the object i'. We use the following distance function, which prefer the larger moving distance for objects other than i':

$D = -min_i \sum_{i=1,i\neq i'}^{m} \sum_{j=1}^{2} (\hat{p}_{T,i,j} - x_{i,j})^2$

4 Results and Analysis

We evaluate our model based on 4 reasoning related to the prediction quality, generalization ability across time unseen environment configurations, and the ability to plan actions for downstream tasks.

4.1 Quality of Predicted Dynamics

To evaluate how well the world dynamics is modeled, we first report the average prediction errors on the test split, over the same time-horizon as which model is trained on, i.e., $t \in [0, T_{train}]$. The prediction error is calculated by the squared l_2 distance between predicted object center location and the ground-truth object centers. The results are shown in Table 1 (left half). Firstly, we show the effectiveness of our proposed RoI Feature by comparing Table 1 VIN, OM, CVP, and Ours (IN). These four entries use the same backbone network and interaction network modules and only visual encoder is changed. Among them, VIN cannot even be trained on PHYRE-W since it cannot handle varying number of objects. The OM method performs slightly better than other baselines since it also explicitly models objects by instance masking. For CVP, it cannot produce reasonable results on all of the datasets except on the SS dataset. The reason is that in PHYRE and billiard environments, cropped images are unaware of environment information and the relative position and pose of different objects, making it impossible to make accurate predictions. In SS, since the object size is large and objects are very close to each other, the cropped image regions already provide enough context. Also, for SS, our implementation gets 5.28×10^{-3} squared l_2 distance for $T \in [0, T_{train}]$ while the original paper [10] report 6.69×10^{-3}. In contrast, our RoI Feature can implicitly encode the context and environment information and it performs much better than all baselines. In the very challenging PHYRE dataset, the prediction error is only $\frac{1}{4}$ of the best baseline. In the other three easier datasets, the gap is not as large since the environment is less complicated, but our method still achieves more than 10% improvements. These results clearly demonstrates the advantage of using rich state representations. Secondly, we show that the effectiveness of our proposed Convolutional Interaction Network by comparing Table 1 Ours (IN) and Ours (CIN). With every other components the same, changing the vector-form representation to spatial feature maps and use convolution to model the interactions can further improve the performance by 10% to approx. 40%. This result shows our convolutional interaction network could better utilize the spatial information encoded in the object feature map.

4.2 Generalization to longer horizon than training

Generalize to longer horizons is a challenging task. In this section we compare the performance of predicting trajectories longer than training time. In Table 1 (right half), we report the prediction error for $t \in [T_{train}, 2 \times T_{train}]$. The results in this setting are consistent with what we found in first analysis in the above section. Our method still achieves the best performance against all baselines. Specifically, for all datasets except SimB, we reduce the error by more than 30%. In SimB, the improvement is not as significant because the interaction with environment only includes bouncing off the boundaries. Meanwhile, changing IN to CIN further improve the performance. This again validates our hypothesis that the key to making accurate long-term feature prediction is the rich state representation extracted from an image.

method	visual encoder	$t \in [0, T_{\text{train}}]$				$t \in [T_{\text{train}}, 2 \times T_{\text{train}}]$			
		PHYRE-W	SS	RealB	SimB	PHYRE-W	SS	RealB	SimB
VIN	Global Encoding	N.A.	2.47	1.02	3.89	N.A.	7.77	5.11	29.51
OM	Masked Image	6.45	3.01	0.59	3.48	25.72	9.51	3.23	28.87
CVP	Cropped Image	60.12	2.84	3.57	80.01	79.11	7.72	6.63	108.56
Ours (IN)	RoI Feature	1.50	1.85	0.37	3.01	12.45	4.89	2.72	27.88
Ours (CIN)	RoI Feature	**1.31**	**1.03**	**0.30**	**2.55**	**11.10**	**4.73**	**2.34**	**25.77**

Table 1: Comparison of the paper's result with other baseline models for longer horizon and unseen configurations

method	visual encoder	PHYRE-C	SS-4	SimB-5
VIN	Global Encoding	N.A.	N.A.	N.A.
OM	Masked Image	50.28	17.02	59.70
CVP	Cropped Image	99.26	16.88	113.39
Ours (IN)	RoI Feature	10.98	15.02	24.42
Ours (CIN)	RoI Feature	**9.22**	**14.61**	**22.38**

Table 2: Comparison of the paper's result with other baseline models for modified dataset for unseen configurations

4.3 Generalization to unseen configurations

As one of the benefits, our method can generalize to novel environments configurations without any modifications or online training. We test such a claim by testing on several novel environments unseen during training. Specifically, we construct 1) simulation billiard dataset contains 5 balls with radius 2 (SimB-5); 2) PHYRE-C where the test tasks are not seen during training; 3) ShapeStacks with 4 stacked blocks (SS-4). The results are shown in Table 2. Since VIN needs a fixed number of objects as input, it cannot generalize to a different number of objects, thus we don't report its performance on SimB-5, PHYRE-C, and SS-4. In the SimB-5 and PHYRE-C setting, where generalization ability to different numbers and different appearances is required, our method reduce the prediction error by 75. In SS-4, the improvement is not as significant as the previous two because cropped image may be enough on this simpler dataset as mentioned above.

4.4 Learned model quality for planning actions

The advantage of using a general purpose task-agnostic prediction model is that it can help us do downstream planning and reasoning tasks. In this section, we evaluate our prediction model in the recently proposed challenging PHYRE benchmark [9] and simulation billiards planning tasks.

PHYRE. We use the B-tier of the environment. In this task, we need to place one red ball at a certain location such that the green ball touches another blue/purple object. [9] trains a classification network whose inputs are the first image and a candidate action, and outputs whether the action leads to success. Such a method does not utilize the dynamics information. In contrast, we can train a classifier on top of the predicted objects' features so that it can utilize dynamics information and makes more accurate classification. During training, we use the same prediction model as described in previous sections except for $T_{train} = 10$, and then attach an extra classifier on the objects' features. Specifically, we concatenate each object's features at the 0th, 3rd, 6th, and 9th timestep and then pass it through two fully connected layers to get a feature trajectory for each object. Then we take the average of all the objects' features and pass it through one fully-connected layer to get a score indicating whether this placement solve the task. We minimize the cross-entropy loss between the score and the ground-truth label indicating whether the action is a success. Note that different from

	Within	Cross
RAND	13.7$_{\pm 0.5}$	13.0$_{\pm 5.0}$
DQN	77.6$_{\pm 1.1}$	36.8$_{\pm 9.7}$
Ours	**85.2**$_{\pm 0.7}$	**42.2**$_{\pm 7.1}$

Table 3: Planning action accuracy for PHYRE dataset based on two score functions RAND and DQN [9]

	Target State Error	Hitting Accuracy
RAND	36.91	9.50%
CVP	29.84	20.3%
VIN	9.11	51.2%
OM	8.75	54.5%
Ours	**7.62**	**57.2%**

Table 4: Planning action accuracy for SimB dataset based on target state error and hitting accuracy

previous work [9], our model does not need to convert the input image to the 7-channel segmentation map since object information is already utilized by our object-centric representation. During testing, We enumerate the first 10000 actions from the pre-computed action set in [9] and render the red ball on the initial image as our prediction model's input. Our final output is an sorted action list according to the model's output score. We report the AUCCESS metric on the official 10 folds of train/test splits for both within-task generalization and cross-task generalization setting. The results are in Table 3. Our method achieves about 8 points improvement over the strong DQN baseline [9] in the within task generalization setting. On the more challenging setting of cross-task generalization where the environments may not be seen during training, our method is 6 points higher than DQN.

SimB Planning. We consider two tasks in the SimB environment: 1) Billiard Target State. Given an initial and final configuration after 40 timesteps, the goal is to find one action that will lead to the target configuration. We report the smallest distances between the trajectory between timestep 35-45 and the final position. 2) Billiard Hitting. Given the initial configurations, the goal is to find an action that can hit the other two balls within 50 timesteps. We firstly train a forward model taken image and action as input, to predict the first 4 object positions and render it to image. After that the rendered images are passed in our prediction model. We score each action according to the similarity between the generated trajectory and the goal state. Then the action with the highest score is selected. The results are shown in Table 4. Our results achieves best performance on all tasks.

5 Conclusion

In this paper, modern computer vision and deep learning techniques were used to propose Region Proposal Interaction Networks for physical interaction reasoning with visual inputs. We also showed that this network and model architecture achieves a significant improvement and can generalize across both simulation and real-world environments for long-range prediction and planning. The authors suggest that this model can be taken as a good benchmark for developing future methods in the field of learning intuitive physics, as well as their application to real-world robotics.

References

[1] Haozhi Qi, Xiaolong Wang, Deepak Pathak, Yi Ma, Jitendra Malik, Learning long-term visual dynamics with region proposal interaction networks. In ICLR, 2021.

[2] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In AAAI, 2017.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. arXiv, 2018.

[4] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. ICCV, 2019.

[5] Ross Girshick, Fast R-CNN. In ICCV, 2015.

[6] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In NeurIPS, 2016.

[7] Frank Allgöwer and Alex Zheng. Nonlinear model predictive control. Birkhäuser, 2012.

[8] Eduardo F Camacho and Carlos Bordons Alba. Model predictive control. Springer Science Business Media, 2013.

[9] Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre:A new benchmark for physical reasoning. In NeurIPS, 2019

[10] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction.ICCV, 2019.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. ICLR, 2014

[12] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In CVPR, 2017

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In ECCV, 2014

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv, 2014

[15] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. arXiv, 2016

[16] Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In NeurIPS, 2017

[17] Michael Janner, Sergey Levine, William T Freeman, Joshua B Tenenbaum, Chelsea Finn, and Jiajun Wu. Reasoning about physical interactions with object-oriented prediction and planning. ICLR, 2019

[18] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In NeurIPS, 2016

[19] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation.In ECCV, 2016

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, 2015

[21] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. ICLR, 2015